Norwegian University of Science and Technology

Master's Thesis

TDT4900

Mixed Attribute Types in Similarity Measures in the Retrieval Step of Case-based Reasoning

Author
Ole Berdal
oleberd@stud.ntnu.no

Supervisor
Prof. Agnar Aamodt, IDI
Co-Supervisor
Ketil Bø, Trollhetta

August 3, 2019

NTNU
Norwegian University of
Science and Technology

Abstract

A novel area of research is the task of computing similarity on heterogeneous attributes, as well as transforming between categorical and numerical values. The research portrayed in this report will address the field of Artificial Intelligence, and more specifically similarity measures in the context of the Retrieval step of Case-based Reasoning. The purpose of this study is to advance understanding of both similarity measures and of transformation methods for attributes of mixed data types in classification problems. The research will naturally have applications outside the scope of Case-based Reasoning as well. Our work will be conducted with a theoretical study supplemented by an applied research experiment in collaboration with Trollhetta, an AI company located in Trondheim. A working demo that employs k-NN is proposed to handle classification of mixed attributes composed of categorical and numerical values. We contribute to the scientific community with an evaluation of optimal strategies when it comes to employing categorical vs continuous similarity measures on a data set with mixed attribute types. A new technique for learning the mutual translation between categorical and numerical values is also introduced.

Acknowledgements

Thank you Agnar Aamodt and Ketil Bø, for giving me a project, and being understanding supervisors—especially with regards to my remote work location. Thank you Anders Kofod-Petersen, for providing me with a template for the thesis. Thank you to those who have proofread it. And especially thanks to you Belén, for your unconditional support and push when I needed it the most.

Ole Berdal
Oslo, August 3, 2019

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter will give an introduction to the research and work to be presented in this master's thesis. Section 1.1 will introduce the problem at hand and the driving forces motivating our research. Section 1.2 will go into details on the goals and research questions we aim to answer. Section 1.3 will lay out the research method used. Section 1.4 will briefly summarize the contributions of the work within the field of Case-based Reasoning (CBR) and the scope of similarity measures. And lastly, Section 1.5 introduces the remainder of the thesis.

1.1 Motivation

The study of our work is derived from the following application-driven problem scenario derived in collaboration with Ketil Bø, the CEO of Trollhetta[1] and co-supervisor to this master's project.

Problem scenario: *Medical personnel operate under varying circumstances and with different resources available to them at any given time. A robust support tool for decision-making for medical diagnosis that can be utilized under various conditions, and with input values of mixed data types such as both qualitative and quantitative data is desired.*

This is a broad initial problem description that can be interpreted in a number of ways. For the purpose of conducting research, we have put emphasis on the

[1]Trollhetta is a software company offering solutions or added functionality to existing systems based on its proprietary development tools for machine vision, artificial intelligence and dynamic modelling. `https://www.trollhetta.com/`

aspect of handling and accepting mixed data types in our input parameters for such a support diagnostic tool. And this is what our research will be centered around. As such, the remainder of this section will be dedicated to further elaborate on our understanding and interpretation of this problem scenario, as well as introduce related work within the scope of our research topic. We start off with a *use case* to exemplify and describe a situation where our work applies.

Use case: *In a thriving welfare state, where the hospitals and healthcare system is organized publicly, under an umbrella organization, there is an ambition to utilize smart support tools for diagnosing breast cancer in patients across geographic location and establishment. This means a unified tool should be equally available in a wealthy, well-equipped hospital as well as in a general practitioner's office with limited medical appliances. That is to say the same system, with the same data available must be able to handle precise numerical input parameters as well as imprecise categorical input parameters, and confidently give a more or less accurate diagnosis.*

From our initial problem description and the above-mentioned use case, on which our work is based, we define the core of our work to be the use of mixed data for the same variable in classification problems. Not only is this uncharted territory for me, but existing research on this topic is rather modest. Hopefully, it is not that the idea behind our work is futile which is the reason existing work on this particular research problem is so scarce. In any case, we hope our work will shed new light on a novel area, or inspire further research in the field of study. As described above, this can have great impact on real-world applications, especially in the future as society and technology advance. An argument might be made that our software tools are better off being specialized for one narrow task rather than accommodate for uncertainties when translating fuzzy categorical concepts into numerical values. For this reason, our main angle will be from a research perspective, and our motivation is hence mostly technique driven rather than application driven. Broadly speaking, our motivation is to enrich the knowledge in the field of artificial intelligence with regards to the aforementioned challenge in the pursuit of technological advancement.

Measuring similarity has many different qualities and uses as is illustrated by Cunningham [2008] in his attempt to organize a taxonomy for similarity mechanisms in CBR. In the simplest and most commonly used sense similarity is a simple distance metric for numerical feature-value data. However, not all data is homogeneous. How can for example the variable *color* be compared if one instance describe said variable textually, another in RGB color code, and a third in wavelength of light waves as seen in Table 1.1. Similarity measures for mixed

Instance	Color	Shape	Weight
Object 1	red	Rectangle	503 g
Object 2	#D01212	Circle	1000 g
Object 3	700 nm	Rectangle	31 g

Table 1.1: Example of different data types for same variable.

data type variables as described above is the essence of what we are going to research. To be able to measure similarity, the domain on which the comparison takes place has to be homogeneous. That is to say that from our previous example with the colors, 'red' cannot be compared to #D01212 directly. They need to be translated into a common representation on the same domain in order to mathematically measure their similarity.

What representation data is stored as can also matter for the outcome of the similarity measure, as well as the process of gathering the data. It might be easier to identify the color as 'red' instead of obtaining the precise measurement of 700 nm (in collecting the data), but the output of the algorithm might yield better results with the 700 nm value because the translation from 700 nm to 'red' loses some information[2]. Therefore, to be able to compute similarity between different data types, we need to investigate the translation – or transformation – between different data type domains. We are also interested in the performance of different similarity measures on different data representations, namely continuous similarity measures and categorical similarity measures—which will both be further explained in Chapter 2.2. Another consideration when attempting to translate between continuous and categorical values is that continuous values are more rigid and exact, while categorical values more often than not have an uncertainty and impreciseness to them. Another way to put this is that categorical values have an underlying abstract meaning to them. This is well illustrated in Figure 1.1 that we have borrowed from Willems et al. [2019].

Our work include a literature study in order to place our work relative to existing scientific research. This was done in Chapter 3. During this part of our project we have learned – due to the lack of existing literature – that the problem we are tackling is a novel one with a lot of potential for further study. We therefore divided our problem into two sub-problems that we researched to the best of our ability in our literature study. We studied papers by Boriah et al. [2008], Cheung and Jia [2013] and Bahari and Van hamme [2014] when researching *Categorical Similarity Measures*; and papers by Harwell and Gatti

[2]Translation from 'red' to 700 nm assumes a generalization of the color red to be approximately the range of light wave wavelengths between 700 and 635 nm.

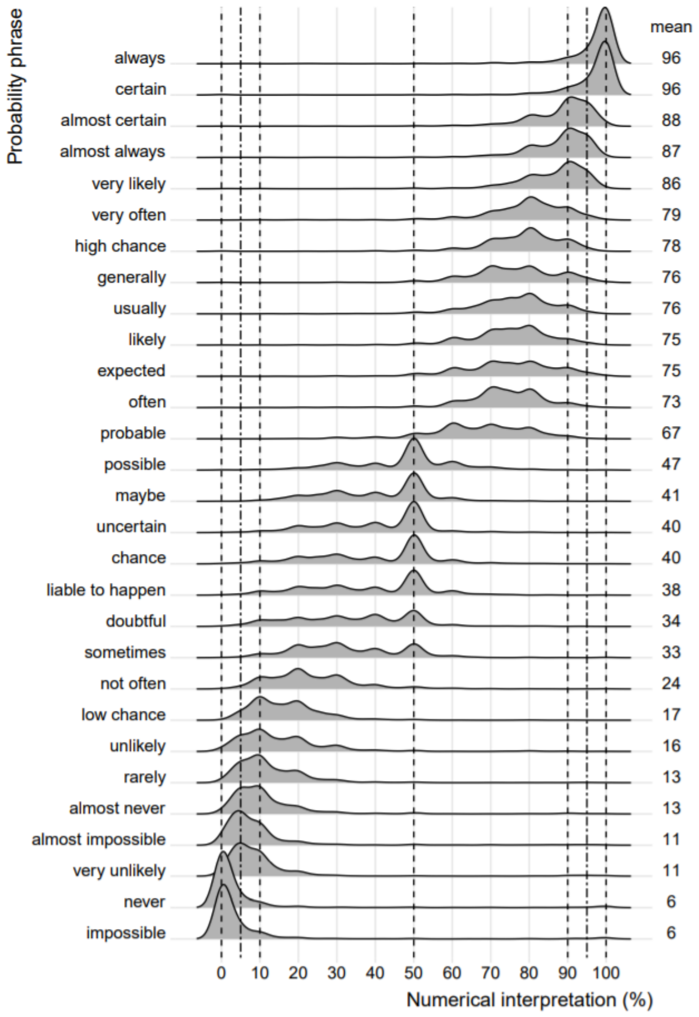


Figure 1.1: Density plots and mean values of the numerical interpretations of categorical values.

Possible object combination	**Similarity policy**
Two quantitative variables	Categorical similarity measure
	Numerical similarity measure
Mixed quantitative and qualitative variables	Categorical similarity measure
	Numerical similarity measure
Two qualitative variables	Categorical similarity measure
	Numerical similarity measure

Table 1.2: Different similarity measure policies depending on variable types.

[2001] and Zdravevski et al. [2015] when researching *Transformation Techniques*. The study of similarity measures was helpful, but our study of transformation techniques bore no fruits for the purpose of our work. This goes to show our use of transformation techniques in our context is a novel one.

1.2 Goals and Research Questions

The overall goal of this thesis is to research efficiency and implications of classification on data with mixed type variables.

Goal *Research and explore implications of classification on data with mixed type variables, and performance with regards to different techniques used for similarity computations.*

There are two main aspects that will be investigated: (1) transformation techniques and (2) the use of categorical vs numerical similarity measures. With respect to transformation techniques we want to explore different options and compare their performance. And with respect to similarity measures we want to compare performance based on the policy of choosing categorical similarity measures vs choosing numerical similarity measures depending on the instances to be compared. The different possibilities of this policy is illustrated in Table 1.2. We will not investigate every possible combination, but rather four of the possible eight combinations. These will be further substantiated later, in Chapter 4.1.

We aim to answer the following Research Questions in line with our goal.

Research Question 1 *What transformation techniques exist to handle input parameters of both quantitative and qualitative form?*

Research Question 2 *How can suitable techniques be implemented in a working demo that accepts and handles input parameters of both quantitative and qualitative form?*

Research Question 3 *How do techniques for using categorical vs numerical measures, based on the training data, compare with respect to performance differences?*

1.3 Research Method

We are addressing our goal and research questions in a few ways. Firstly, we do a light theoretic background study to investigate different technologies and methods that already exist out there in the vast sea of knowledge and information. Upon having gathered sufficient useful information, we will develop a demo to perform analytical tests and present our findings in a scientific manner. This methodology is chosen to introduce a novel area of similarity research into the scientific community in a modest way. The architecture and design of our experiments and demo will be introduced in Chapter 4.1.

In terms of researching similarity, CBR is a prime platform to conduct our experiments. Similarity measures arise naturally in one of the core steps of the CBR cycle—Retrieval[3]. And since CBR is divided into separate steps, we will be able to deploy tests on the relevant step in isolation and thus circumventing unnecessary implementation of a full-scale application. With regards to our problem scenario and use case, CBR also makes sense as it is a methodology where reasoning and transparency is vital. That is to say a user of the tool must not blindly accept the output of the tool as if it were a *black box*[4], but needs to see the reasoning behind the decision-making. Unarguably, it is an important point for medical personnel to be able to confidently utilize the output of the tool in situations where human lives are on the line. This question is raised in this CNN Business article [Lewis, 2019] that refers to another study on classifying breast cancer – but with a deep convolutional neural network – by Yala et al. [2019].

To conduct our experiments and perform our tests we will utilize the programming language Python. Python is both a strong and dynamic language with high readability, as well as strong support and commonly used in scientific and research environments. For our data set, we will use a publicly available open data set from the UCI Machine Learning Repository [Wolberg, 1992]. This data set is a collection of 699 instances with classifications of breast cancer occurrences in patients. No personal patient information is attached to the data, and the data is already normalized and complete. As all data in the data set is numerical we will prepare the data set to simulate mixed data types. This will

[3]The life cycle of CBR is described generally by Aamodt and Plaza [1994] to have four REs (RETRIEVE, REUSE, REVISE, RETAIN), whereas retrieve fetches the most similar cases to the problem case from the case base. More on this in Chapter 2.1.2.

[4]In science, a black box is a closed system in which you can see the inputs and the outputs, but it is not known what is going on inside the system.

be explained in detail in Chapter 4.2.

1.4 Contributions

This section provides a brief summary of the main contributions of the work. The contributions in their entirety is disclosed and fully listed in Section 6.1. Below we give the shortened version of our contributions:

1. *Introduction of a novel research area to the scientific community.*
2. *A working demo for a classifier on heterogeneous attributes.*
3. *Comparison of continuous vs categorical similarity measures on mixed data.*

1.5 Thesis Structure

In this section we provide the reader with an overview of what is coming in the next chapters.

Introduction In this chapter we have given an brief introduction to the work we are presenting in this master thesis. Our motivation, goal and research questions, our research method, and contributions have been presented.

Background Theory This chapter introduces relevant terminology, as well as establishing relevant topics related to our study. Case-based Reasoning, k-Nearest Neighbor, Model Evaluation, both Continuous and Categorical Similarity Measures, Transformation Techniques, and Fuzzy Logic are introduced.

Related Work In Chapter 3 we detail the Structured Literature Review Protocol used to find existing relevant literature. Relevant literature with respect to similarity measures, and transformation techniques are reviewed and summarized.

Methodology In this chapter we declare our plan driving the experimental research in the first half, and in the second half the experimental setup is revealed together with the introduction of our data.

Results and Discussion In Chapter 5 we present and visualize the results, we evaluate the results, share our main findings, we share our thoughts and discuss regarding merits and limitations of our work.

Conclusion Finally, in Chapter 6 we conclude our thesis by disclosing the contributions of our work, as well as suggest future work in our field.

Chapter 2

Background Theory

In this chapter we will go through various terminology and concepts that will provide the reader with a deeper understanding and greater intuition when reading through the rest of this document. In Section 2.1 there will be an introduction to useful terminology, a brief summary of what Case-based Reasoning is, and a presentation of k-Nearest Neighbor—the method we use in our supervised classification problem. In Section 2.2, Section 2.3, and Section 2.4 we will adequately introduce similarity metrics, transformation techniques and fuzzy logic, respectively.

2.1 Machine Learning

Machine Learning (ML) is the scientific study of algorithms and statistical models that computer systems use in order to perform a specific task effectively without using explicit instructions, relying on patterns and inference instead. Traditionally, the difference between human and computer has been our ability to learn and adapt to new information and new problem scenarios. Until recently, most computers have been explicitly programmed to perform narrow and specific tasks. Although the term Machine Learning has been around for a long time, it is not long ago since a surge in activity and popularity around Machine Learning has happened. Even if we are not conscious about it, data on us is continuously processed by Machine Learning systems on a daily basis to influence us, get a hold of our attention, and ease our lives. Entertainment services like Spotify and Netflix are recommending personalized content based on our engagement history and interactions with their platforms. We are so lucky to live in a world with free access to content and information, but is it really free? We are constantly bombarded with tailored advertisements that ultimately influence our purchases and fuels consumerism. This personalized advertisement is of course also powered by machine learning systems. We could keep going on and on about this, but the point is that Machine Learning has become inevitable. Machine Learning is increasingly present in our everyday lives, but it was not more than 60 years ago the name *machine learning* was coined by Samuel [1959] in his article where he investigated machine learning procedures in the game of checkers. He described it as:

"The field of study that gives computers the ability to learn without being explicitly programmed."

Some 40 years later Mitchell [1997] provided the widely quoted, and more formal definition below:

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

In this section we will introduce the various relevant Machine Learning terms and concepts that will be used throughout the remainder of the thesis.

2.1.1 Terminology

Some of the terms defined here are used unambiguously in the field of machine learning, while others are used in several, often related ways. The definition given here is how the term should be interpreted for the rest of the thesis.

Features/Attributes These are the data points from the data instances used by our algorithm. These can be used interchangeably. These features takes on discrete numerical values or categorical values.

Parameters The variables we manually tune in order to get a accurate model. These can be chosen by different heuristics like static analyses, or trial and error. The parameters for k-NN are for instance the k-value, the number of neighbors we consider, or our distance functions.

Supervised Learning This is a kind of learning where both the data variables and the classes are known to the learner during training. That is to say both input and output are known, and the learnt model learns a function that maps from input to output. The other types of learning are called *Unsupervised Learning* and *Reinforcement Learning.* [Norvig and Russell, 1994, p. 695]

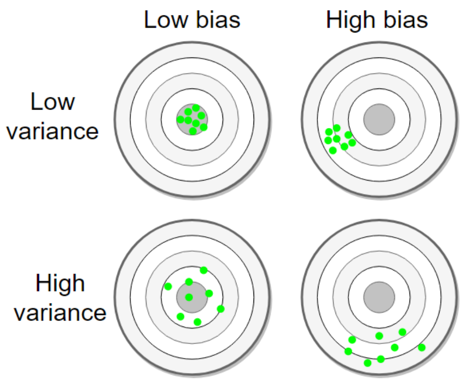


Figure 2.1: Bias-variance tradeoff illustrated.

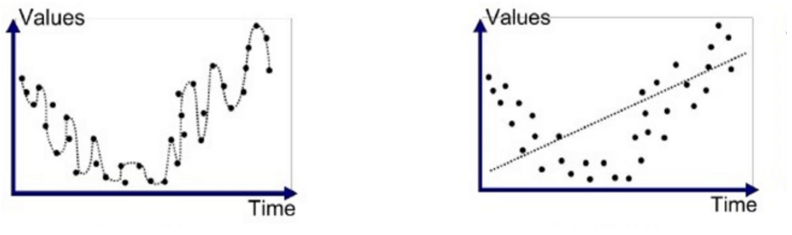
Variance Variance, in the context of Machine Learning, is a type of error that occurs due to a model's sensitivity to small fluctuations in the training set. This can be seen illustrated in Figure 2.1. [Mitchell, 1997, p. 129]

Bias The bias is an error from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to miss the relevant relations between features and target outputs. This can be seen illustrated in Figure 2.1. [Mitchell, 1997, p. 129]

Overfitting If the model fits the data set too well, and fails to map the general pattern of data it is said to overfit. Data with a lot of *variance* is sensitive to this. Too few data instances can also influence if the model might overfit. In k-NN, overfitting is usually connected with having a too small k-value. Overfitting is depicted in Figure 2.2a. [Norvig and Russell, 1994, p. 705]

Underfitting Underfitting occurs when a statistical model cannot adequately capture the underlying structure of the data. An underfitted model is a model where some parameters or terms that would appear in a correctly specified model are missing. This is to say they have a high *bias*. In k-NN, underfitting is linked with a very high k-value. Underfitting is illustrated in Figure 2.2b. [Norvig and Russell, 1994, p. 709]

Leave-One-Out Cross-Validation This approach leaves one data point out of the training data, i.e. if there are n data points in the original sample, then $n-1$ samples are used to train the model and one point is used as the validation



(a) Overfitting *(b) Underfitting*

Figure 2.2: Illustrating overfitting and underfitting in context of regression.

set. This is repeated for all combinations in the original sample set, and the error is averaged across all trials to give performance effectiveness. [Han et al., 2012, p. 253]

Prediction/Classification This is the problem of predicting to which discrete category or class a given observation belongs. This can also be referred to by *target value.*

2.1.2 Case-based Reasoning

Case-based Reasoning [Richter and Weber, 2013] is a machine learning methodology first brought to light by Schank [1983] when he proposed the dynamic memory theory. For more than three decades CBR has been a flourishing field that has attracted researchers, practitioners and entrepreneurs alike. It is the process of solving new problems based on the solutions of similar past problems. The CBR methodology aim to provide a computational model that is very close to human reasoning. The name consists of three words deserving a short explanation. A **case** is basically an experience of a solved problem, and a case base is a collection of such cases. The term **based** merely emphasizes that the cases are the original source for the reasoning. **Reasoning**, the term most characteristic of the approach, means that the system shall draw conclusions using cases. It encompasses four famous steps known as the four **R**s introduced by Aamodt and Plaza [1994], and further conceptually formalized and specified by Stahl [2005]. These four Rs are *retrieve*, *reuse*, *revise* and *retain.*

1. **Retrieve** fetches the most similar case or cases.

2. **Reuse** takes the information and knowledge in that case and uses to solve the problem.

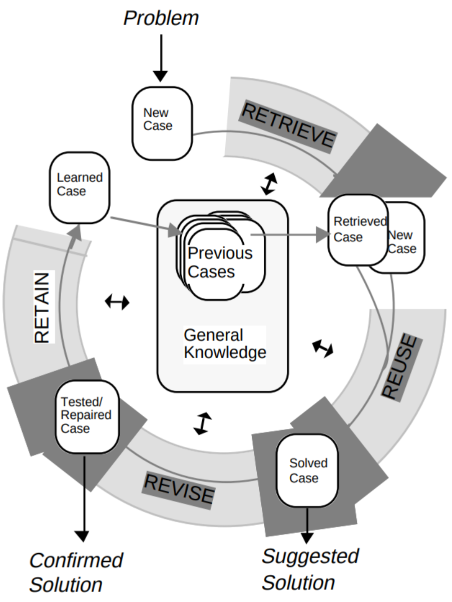


Figure 2.3: The Case-based Reasoning cycle.

3. **Revise** reviews the proposed solution.
4. **Retain** incorporates the parts of the experience likely to be useful for future problem solving.

A new problem is solved by retrieving one or more previously experienced cases, reusing the case in one way or another, revising the solution based on reusing a previous case, and retaining the new experience by incorporating it into the existing case base. Aamodt and Plaza [1994] illustrates this cycle in Figure 2.3.

2.1.3 k-Nearest Neighbors

The k-Nearest Neighbor (k-NN) [Mitchell, 1997, p. 231] is a widely used machine learning algorithm. It is an instance-based – also known as lazy – learner. This means it does not generalize a model based on the data before receiving a query, but instead compares new problem instances with the instances stored in memory. The algorithm assumes all instances correspond to points in the n-dimensional space. It classifies a new instance by computing the distance between the new instance with all other known instances, and uses the k nearest neighbors to determine the predicted classification for the new instance. k-NN is advantageous in that it is very intuitive and simple to implement, there is no training step, it responds quickly to incorporating new data. But on the flip side it is slow compared to *eager learners*[1], it scales badly when the sample size increases, and is sensitive to outliers. The algorithm is typically executed with numerical values, and the distance function is typically defined to be $d(x_i, x_j)$:

$$d(x_i, x_j) \equiv \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2} \tag{2.1}$$

where $\mathrm{a_r(x)}$ denotes the value of the r^{th} attribute of instance x. It is also possible to extend whatever distance function to also *weigh* the attributes according to importance as shown below:

$$d(x_i, x_j) \equiv \sqrt{\sum_{r=1}^{n} w_r d(a_{ir}, a_{jr})} \tag{2.2}$$

where $d(a_{ir}, a_{jr})$ symbolizes the distance between attribute $\mathrm{a_i}$ and $\mathrm{a_j}$ for attribute r according to the chosen distance metric, and w_r is the weight of attribute r such that:

$$0 \leq w_r \leq 1 \quad \text{and} \quad \sum_{r=1}^{n} w_r = 1 \tag{2.3}$$

Upon collecting all k nearest neighbors, the algorithm classifies the new instance based on the these closest data instances given a predetermined voting method. This voting method can be a simple majority voting where the class of the majority of the k neighbors are chosen. Table 2.1 is a table of the voting methods we experiment with in our implementation. In the table, *distance* is the actual distance in question, $distance_{max}$ is the maximum possible attribute distance in the data collection according to the distance function, $distance_{nearest}$ is the distance between the nearest instance and the new instance, likewise $distance_{farthest}$ is the distance of the farthest instance (in the selected k nearest neighbors), and i is the i^{th} nearest neighbor in k.

2.1.4 Model Evaluation

An important part of Machine Learning is being able to evaluate the performance of the model. There are several reasons why the model evaluation is extremely

[1]An eager learner is a learning method which tries to construct a general, input-independent target function during training of the system, as opposed to *lazy learners*.

method	**formula**
majority	1
inverse	$\frac{1}{\frac{1}{distance_{max}}+distance}$
logarithmic	$\frac{1}{1+distance}$
uniform	$\frac{1}{i}$
standard	$\frac{distance_{farthest}-distance}{distance_{farthest}-distance_{nearest}}$
new_standard	$\frac{1}{i}\frac{distance_{farthest}-distance}{distance_{farthest}-distance_{nearest}}$

Table 2.1: The different voting methods we use up against each other.

useful when working with machine learning algorithms. The ones we feel are most important are listed below:

- Estimating the performance is helpful for picking the best learning algorithm for the problem.
- Tuning the parameters to best classify unseen data.
- Estimate predictive performance when the model performs classification on unseen data.

In any case, the most important aspect of model evaluation is determining the actual usefulness and performance of the model on unseen data. It is undesirable to use the same data to train the model and to test the model. A model that is given the same data for training and for testing purposes is likely to end up unreliably being evaluated as performing really well. If this is not the case, either there is likely some problem with the data, the algorithm might fail to capture the pattern of the data, or there might not be any inherit pattern to learn in the data. Anyhow, the desired action is to test the model on unseen data. There are various ways to do this. The most simple and sensible way is perhaps the *Hold-Out* method. This method is best used on large data sets, and the data set is randomly divided into two or three subsets (depending on the model we are evaluating):

1. **Training set** is the subset in which the model is trained on to improve its predictive nature.
2. **Validation set** is a subset of the data set used to assess the performance of the model built in the training phase. It provides a platform for fine tuning of the model's parameters, and selecting the best performing model. *Not all modeling algorithms need a validation set.*

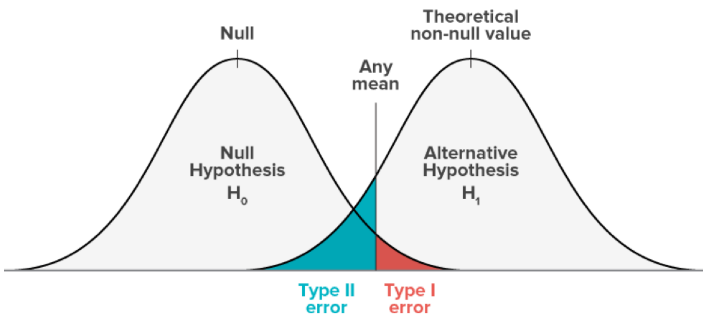


Figure 2.4: Type-I and Type-II errors.

3. **Test set** is the subset used to test the final and ultimate performance of the model on data instances not seen during training. If a model performs much better on the training set than on the test set, then that is a strong indication the model is overfit.

The other method for evaluating models is *Cross-Validation*. This model is more commonly used when the data set is not sufficiently big to split into subsets. Cross-Validation is also known as k-fold cross-validation because the data is divided into k subsets of equal size. It then builds k models, each time leaving one of the subsets out for testing. If k equals the sample size it is called "leave-one-out".

There are two types of errors whose importance depends on the application of the classification algorithm. These errors are simply called *type-I* error and *type-II* error. Type-I errors occur when the *null hypothesis* is rejected when it in fact should not be, and type-II errors occur when the *alternate hypothesis* is true, but the null hypothesis was not rejected. This is analogous to the story about the boy that yelled *"wolf, wolf"*. The first time the boy yelled, the villagers came to his rescue even though there actually was no wolf. The villagers mistakenly believed him. The second time he yelled there actually was a wolf there, but the villagers, this time again, mistakenly did not believe the boy. The balance between these two types of errors is depicted in Figure 2.4. The interesting thing about this balance is that you have to make a deliberate design choice regarding what type of error is more important to reduce. Sometimes one type of error is more important to minimize, but doing this will result in an increase in the other type of error. For instance is it preferred to reduce the number of ill patients that are not classified as being ill (type-I error), even though this increases the number of healthy patients classified as ill (type-II error).

For model evaluation there are many different performance metrics commonly

		prediction outcome		
		p	**n**	**total**
actual value	**p′**	true positive	false negative	P′
	n′	false positive	true negative	N′
	total	P	N	

Table 2.2: A confusion matrix for classification of negatives and positives.

used. Many of them can be understood in relation to the *confusion matrix*, which in itself can also be considered a good measure for model performance. Type-I and type-II errors can also be interpreted from this matrix. Table 2.2 shows the confusion matrix for the common 2-class classification problem of positives and negatives. The four different boxes correspond to the following:

- **tp** = *true positives*, the number of correctly classified positives.
- **tn** = *true negatives*, the number of correctly classified negatives.
- **fp** = *false positives*, the number of erroneously classified positives.
- **fn** = *false negatives*, the number of erroneously classified negatives.

Following these terms, we will define the performance measures we have used in our thesis.

Accuracy is the percentage of correctly classified instances across all classes, negatives or positives. In the definitions of possible classifications of the confusion matrix in Table 2.2 it corresponds to the following:

$$accuracy = \frac{tp + tn}{tp + tn + fp + fn} \tag{2.4}$$

And for the general case of classification with any number of classes the following formula, where $\hat{y}_i$ is the actual class for data point i; y_i is the predicted class for data point i; and n is the number of data points, gives the accuracy:

$$accuracy = \frac{1}{n}\sum_{i}(y_i = \hat{y}_i) \tag{2.5}$$

This is a very natural performance measure to use, but in general is not sufficient on its own as a measure of a models performance. If one class is severely over-represented it can correctly classify that class near 100 percent of the time, but still be useless. One way to determine this is to compare the model accuracy to the performance of a model that always classifies the most prominent class. Just imagine an algorithm that classifies fatal illness that only 0.1% of the population has. An algorithm that classifies 99% of the illnesses and 5% of the healthy as having the illness would have about 95% accuracy, but an algorithm that only classifies as healthy would have a 99.9% accuracy. But it is pretty clear the algorithm with 99.9% accuracy would be completely useless, while the model with 95% accuracy would have singled out a group for further testing.

Precision is a performance measure that might be more useful in the previous example where one class might be underrepresented, but also actually the crucial class to correctly classify. In terms of the confusion matrix in Table 2.2 the precision is the ratio of correctly classified positives over all classified positives.

$$precision = \frac{tp}{tp + fp} \tag{2.6}$$

For the general case of any number of classes we can only calculate the precision of one class at the time. The precision being all correct classification of that class divided by all classifications of that class.

Recall , also called *sensitivity*, is in relation to Table 2.2 the percentage of correctly classified positives over all actual positives. This corresponds to:

$$recall = \frac{tp}{tp + fn} \tag{2.7}$$

Just like with precision we can only calculate the recall of one class at the time in the general case. The recall is calculated by true classifications of that class divided by the number of actual samples of that class. In the two class problem in Figure 2.2 the sensitivity of the negative class is called the *specificity*.

F-measure , or *f₁-score*, is the combination of precision and recall. For the two-class problem illustrated in Table 2.2 this corresponds to:

$$F_1 = 2\frac{precision * recall}{precision + recall} \tag{2.8}$$

It is the harmonic mean between the precision and recall, and like the three previously mentioned performance metrics has values ranging from 0 to 1. The f_1-score is a value that punishes skewed classifications. The f_1-score can be considered a measure for how *balanced* the classification is.

2.2 Similarity Metrics

Similarity metrics, similarity measures, distance, and proximity measures are different terms used to describe similar things. When we want to compute the distance between two instances, the concept of distance depends entirely on the data we work with. Different types of data have different ways to calculate the similarity or distance. Intuitively, we consider *similarity* to be the logical inverse of the distance when we are required to compute this, although we use some tricks to circumvent running into *division by zero* and *infinity* values. Data attributes can be categorized in many different ways, some of which that are relevant and will be mentioned here.

Categorical attributes are categorically discrete attributes that consists of names only, and are finite in numbers. There is no real distance between them, and they are abstract concepts with a fundamental underlying meaning behind them that we humans might be able to infer distance from. Traditionally, the term similarity is used to determine if two categorical instances are equal or not. There are two types of categorical values:

- **Nominal attributes** have no inherent ordering to them, and only differ in being similar or not ($=$, $\neq$). Consider a simple example of weather categories. We can have many different categories in this particular scenario without any concept or notion of order (windy does not always occur before sunny nor is it smaller or bigger than sunny). Similarly movie, music and video game genres, country names, food and cuisine types are other examples of nominal categorical attributes.
- **Ordinal attributes** are attributes that on the other hand do have a natural order to them, but we cannot directly define a distance between them. For instance can the *height* variable have the following values: 'tall', 'average', and 'short'. The distance can not be given in number (without any sort of method to *learn* this). We can apply $<$, $>$, $=$, $\neq$ to them.

Numerical attributes are *real numbers*, $\mathbb{R}$, where a meaningful distance can sensibly be computed between different values. The following operations (not limited to only these) can be performed on them: $<$, $>$, $=$, $\neq$, $+$, -.

Another way to categorize attributes is between *qualitative* and *quantitative* data.

Qualitative attributes are categorical attributes usually expressed as category names by means of natural language. They can have order or no order between their values. Nominal and ordinal attribute-data are its two types depending on if its values can be ordered.

Quantitative attributes are expressed as numerical values. They describe the value as a measurable quantity. This value can be exactly measured in terms of numbers. However all numbers are not measurable like the social security number, therefore only measurable attributes are called quantitative attributes.

Lastly, we will introduce *homogeneous data* and *heterogeneous data.*

Homogeneous data is data in which all the attributes are of the same type. For instance all attributes are numerical value types.

Heterogeneous data (or *mixed data*) is data where the attribute data types are not the same. For instance the attribute *velocity* can be given with both numerical values and categorical values like 'fast' or 'slow'.

Following we will introduce well-known methods to compute similarity or distance among the various kinds of attribute categorizations and data types.

2.2.1 Continuous Similarity Measures

Continuous similarity measures are measures we define for computing the distance or similarity of numerical (quantitative) attributes. As mentioned before, this computation is trivial because of the numerical nature that allows for mathematical calculations. The most common continuous similarity measures, or rather distance metrics, are listed below:

Minkowski distance This distance is a metric in a vector space. It can be considered the generalization of the Euclidean/Pythagorean distance and the Manhattan distance.

The Minkowski distance of order p between two points:

$$X = (x_1, x_2, ..., x_n) \quad \text{and} \quad Y = (y_1, y_2, ..., y_n) \tag{2.9}$$

is defined as:

$$d(X,Y) = \left(\sum_{i=1}^{n} | x_1 - y_1 |^p \right)^{\frac{1}{p}} \tag{2.10}$$

An important observation is that the bigger the power p is, the more are big attribute differences reflected in the final distance. In this thesis Minkowski$_1$ distance (Minkowski distance of order 1) is used interchangeably with Manhattan distance, and Minkowski$_2$ distance (Minkowski distance of order 2) is used interchangeably with Euclidean distance.

2.2.2 Categorical Similarity Measures

Inherently, categorical values have no *difference* between values in terms of quantitative numerical values. Usually, they do not even have an ordering to them and even if they do, finding the distance or quantitative similarity might not be easier than for nominal categorical values. In literature there are many attempts to improve categorical similarity measures [Cunningham, 2008], but they turn out to be highly specific for the application and the actual data set in question. It is intuitively easier to define *similarity* instead of distance for categorical values. For our intents and purposes we do not find any issue with this as translation between these two terms can be done fairly easily. More on this in Chapter 4. Below are the different categorical similarity measures we have considered, experimented with, and worked with.

Overlap similarity is a similarity measure that comes from the *overlap coefficient* below, where X and Y are sets.

$$overlap(X,Y) = \frac{| X \cap Y |}{min(| X |, | Y |)} \tag{2.11}$$

Simplified, this is the overlap similarity at the attribute level:

$$similarity_{overlap}\big(x_i, x_j\big) = \sum_{r=1}^{n} sim\big(a_r(x_i), a_r(x_j)\big) \tag{2.12}$$

where $\mathrm{a_r(x_i)}$ is the r^{th} attribute of instance $\mathrm{x_i}$, and $\mathrm{a_r(x_j)}$ equivalently for instance $\mathrm{x_j}$, n the number of attributes, and the similarity function *sim($a_r(x_i)$, $a_r(x_j)$)* is:

$$sim(a_r(x_i), a_r(x_j)) = \begin{cases} 1, & \text{if } \mathrm{a_r(x_i) = a_r(x_j)} \\ 0, & \text{otherwise} \end{cases} \tag{2.13}$$

This is the logically and intuitively most sound similarity measure. It is simple, yet elegant and nicely incorporates the concept of being equal.

2.3 Transformation Techniques

While a lot of advancements have been made in various machine learning frameworks to accept complex categorical data types like text labels. Typically any standard workflow in Machine Learning involves some form of transformation of these categorical values into numeric values. In our particular case we are able to also work with categorical values, but as we aim to compare performance based on using categorical or continuous similarity/distance measures we need some way to transform, translate, or map values from one domain into the other. We are going to employ an educated incremental *trial-and-error*-based method to learn a mapping structure between our categorical and numerical values. We call this the *Incremental Step* method. The learning steps are as listed below:

1. Initiate a categorical-to-numerical mapping table with an arbitrary value. We chose the mean of the lowest and highest possible value which is 5.0 for every attribute.
2. For each location in the (categorical value of each attribute) in the categorical-to-numerical mapping table:
   (a) Run the classification algorithm on the data, translating categorical values into numerical values and measuring distance using a continuous similarity measure.
   (b) Incrementally change the value in the mapping table in either directions and rerun the classification until a peak performance is found.
   (c) Repeat for each location in the mapping table.
3. Next initiate an empty numerical-to-categorical mapping table.
4. For each location in the numerical-to-categorical mapping table:
   (a) Go to the current attribute in the categorical-to-numerical mapping table and read all mapping translations.
   (b) Whichever categorical value that has the translation with the shortest difference is the categorical value for the numerical value in the numerical-to-categorical mapping table.

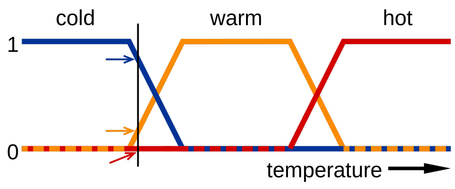


Figure 2.5: Fuzzy logic temperature.

Note that step (3) - (4) can be done in an ad-hoc[2] fashion in the case of continuous numerical values.

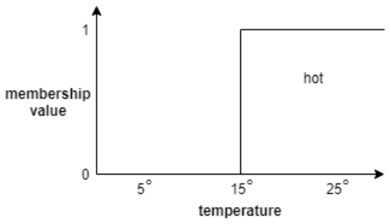
Although our proposed transformation method is only conceptually described above, the actual implementation can be inspected in Appendix B. This is an early working prototype of the method that has a few downsides and limitations, but it gets the job done.

2.4 Fuzzy Logic

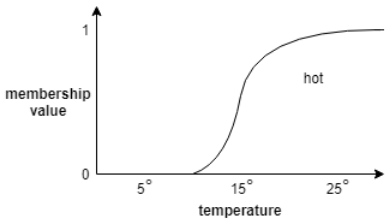
The term *fuzzy logic* was first introduced in a paper on fuzzy set theory by Zadeh [1965]. It is based on the observation that people make decisions based on imprecise, non-numerical and subjective information. They are meant to mathematically capture vagueness and imprecise information—hence the term *fuzzy*. This entails that there are no clear limit between adjacent classes, but rather a gradual and fuzzy transition between classes. This can be seen in the illustration of a fuzzy categorization of temperature in Figure 2.5 we borrowed from Wikipedia [2019]. In this figure it is evident that the terms are not mutually exclusive, and you can have partial membership of multiple classes.

Fuzzy logic is a form of many-valued logic in which the truth values of variables may be any real number between 0 and 1 inclusive. It is employed to handle the concept of partial truth, where the truth value may range between completely true and completely false. By contrast, *Boolean logic* has truth values that may only be the integer values 1 or 0 as illustrated in Figure 2.6. In 2.6a the membership of the 'hot' class is 0 until 15° where it jumps to 1, it is either one or the other. In 2.6b the membership of the 'hot' class is more fluid and gradual, it is more on par with the real world.

[2]When necessary or needed. Ad-hoc is a Latin phrase meaning literally "for this".



(a) Boolean logic



(b) Fuzzy logic

Figure 2.6: Membership functions for Boolean and Fuzzy logic.

Chapter 3

Related Work

In this chapter we will tie existing scientific and academic literature to our research area. In Section 3.1 we will unfold our Structured Literature Review Protocol to disclose how and where we acquired majority of the related articles. Section 3.2 will review relevant literature related to Similarity Measures, and Section 3.3 will visit somewhat relevant literature related to Transformation Techniques.

3.1 Structured Literature Review Protocol

The purpose of this Structured Literature Review (SLR) protocol is to help organize the search for existing literature that are relevant to our research questions. It is not a guarantee of finding *all* relevant literature, but there are several advantages to it anyhow. It can map out existing solutions before the researcher attempts to reinvent the wheel, it helps avoid bias, and it also benefits the community by placing the current study relative to existing research; helping identify gaps of knowledge; and it highlights where additional research is required.

The scope of our search for literature and existing relevant research is summed up in Table 3.1. More articles than is included in this chapter have been studied and explored, however only the ones considered most relevant are included here. Initially we were looking for scientific literature related to similarity of heterogeneous attributes, but we must concede that we were unable to find existing scientific work related to this overall research topic. Instead we divided our search into two sub-searches: *similarity measures* and *transformation techniques*—described in Table 3.1. When we used the *search terms* in our searches, we also tried combinations of the synonyms listed in the *synonyms* column. When we performed the searches we also used different conjugations for the terms. Moreover,

search terms	RQ	synonyms	search engine
similarity measures	RQ3	(categorical ∨ mixed ∨ ordinal ∨ qualitative) ∧ (distance ∨ dissimilarity) ∧ (metric)	Google Scholar, CORE
transformation techniques	RQ1	(ordinal ∨ numerical ∨ categorical ∨ quantitative ∨ qualitative) ∧ (translation ∨ mapping) ∧ (methods ∨ functions)	Google Scholar, CORE

Table 3.1: Systematic literature review protocol.

most articles were found using the structured literature review protocol, but additional articles were introduced to us by other means such as recommendations by supervisors or found in article references.

3.2 Similarity Measures

The notion of similarity for continuous data is relatively well-understood, but for categorical data, the similarity computation is not straightforward. It is of utmost interest to us to calculate similarity between categorical data instances. In this section we will summarize related work that has done this to some extent.

In their paper on a comparative evaluation of similarity measures for categorical data, Boriah et al. [2008] employ 14 different data-driven categorical similarity measures and compare their results on 18 different data sets. They aim to evaluate the performance of these different similarity measures relative to each other in the context of *outlier detection*. From this paper we will consider various categorical similarity measures and investigate whether any can be applicable to our efforts. In short, this article investigates variations of the *overlap* similarity with regards to frequency distribution and other statistical observations. Overlap similarity is criticised in being too simplistic by giving all matches and mismatches equal importance. They differentiate the various similarity measures into three groups: (1) *diagonal entries only*, (2) *off-diagonal entries only*, and (3) *both diagonal and off-diagonal entries*. They finally conclude that they all perform better in their own particular applications, and they make the observation that those who often perform well in one application and bad in another have counterparts with the opposite performance. Picking the ultimate categorical similarity measure is an optimization problem in itself, but nevertheless this paper can serve as inspiration for future work for our project.

Next we take a look at an article that delves into the territory of a unified similarity metrics for clustering on mixed data [Cheung and Jia, 2013]. They note it is nontrivial to perform clustering on mixed data composed of categorical and numerical attributes, and continuing with that there is an awkward gap between the similarity metrics for categorical and numerical data. In their paper they propose a unified clustering approach for both categorical and numerical attributes. Their unified approach is simply a formalization of the categorical similarity in the specific case of clustering where categorical occurrences are counted, and combined with normal continuous distance measure. Their solution solves their case elegantly, but in the context of our problem they lack any regard to mixed attributes where the same variable is constituted of both categorical and numerical values.

A normalized ordinal distance for classification problems are introduced by Bahari and Van hamme [2014]. They look at existing application-driven solutions and come up with a new application-independent performance metric for ordinal classification problems—probabilistic-ordinal and partial-ordinal. They first introduce the ordinal distance between arbitrary vectors in Euclidean space, before proposing a new normalized ordinal performance metrics based on the introduced ordinal distance. This new performance metric is conceptually simple, computationally inexpensive, and application-independent. They cover five existing application-dependent solutions and note their shortcomings and disadvantages. They look at *Mean Zero-One Error*:

$$E_{mzo} = \frac{1}{M} \sum_{m=1}^{M} 1_{\hat{y}_m} \neq y_m \tag{3.1}$$

where M is the total number of test set data points, $\hat{y}_m$ is the predicted label of the m^{th} test set data point and y_m is the true label of the m^{th} test set data point. They praise it for its simplicity, but note that it does not suffice because it doesn't consider the order of the categories.

Mean Absolute Error of Consecutive Interger Labels (E_{MA}^{CIL}) that transforms both true and predicted labels into consecutive integers so that the d^{th} column of the label vector is 1 means then the transformed label is equal to d:

$$\mathrm{E}_{MA}^{CIL} = \frac{1}{M} \sum_{m=1}^{M} \mid \hat{U}_m - U_m \mid \tag{3.2}$$

where $\hat{U}_m$ is the transformed predicted label of the m^{th} test set data point and U_m is the transformed true label of the m^{th} test set data point. They note this approach takes the order of categories into account, however it cannot be applied to evaluate ordinal classification problems nor is the range of output application-independent.

They also look at *Percentage of Correctly Fuzzy Classified Instances (P_{CFCI})* that has been applied to fuzzy classifiers. It is calculated as follows:

$$P_{CFCI} = \frac{100}{M} \sum_{m=1}^{M} (1 - \frac{1}{2} \sum_{d=1}^{D} \mid \hat{y}_{m,d} - y_{m,d} \mid) \tag{3.3}$$

But they note that it can be inferred from the above equation that the order of categories is not considered here either.

Their fourth investigation is *Average Deviation (E_{AD})* that evaluates classifiers in fuzzy ordered classification problems, calculated as follows:

$$E_{AD} = \frac{1}{M} \sum_{m=1}^{M} \sum_{d=1}^{D-1} \mid \sum_{i=1}^{d} \hat{y}_{m,i} - \sum_{i=1}^{d} y_{m,i} \mid \tag{3.4}$$

This also suffers from being application-dependent and hence difficult to interpret.

Lastly they review *Averate Ranked Probability Scores (E_{RPS}).* A ranked probability to score the output probabilistic classifiers, defined as follows:

$$RPS_Y(\hat{Y}) = \frac{1}{D-1} \sum_{d=1}^{D-1} (\sum_{i=1}^{d} \hat{y}_i - \sum_{i=1}^{d} y_i)^2 \tag{3.5}$$

and extended to measure performance of classifiers in ordinal classification problems, partial-ordinal classification problems and probabilistic-ordinal classification problems using the following relation:

$$E_{RPS} = \frac{1}{M(D-1)} \sum_{m=1}^{M} \sum_{d=1}^{D-1} (\sum_{i=1}^{d} \hat{y}_{m,i} - \sum_{i=1}^{d} y_{m,i})^2 \tag{3.6}$$

They note this approach may lead to erroneous interpretations due to a weak conservative assumption that that the maximum of the nominator of E_{RPS} is $M(D-1)$.

Their proposed *Normalized Ordinal Distance (E^P_{NOD})* performance metric, based on the notion of the *Minkowski distance of order p* as introduced in Section 2.2.1 is defined as:

$$E^P_{NOD} = \frac{\sum_{m=1}^{M} \mid\mid Y_m - \hat{Y}_m \mid\mid_p^{OD}}{\sum_{m=1}^{M} \Psi^p_{Y_m}} \tag{3.7}$$

where $\Psi^p_{Y_m}$ is the upper bound of $\mid\mid Y - \hat{Y} \mid\mid_p^{OD}$ for any possible $\hat{Y}$ in its defined range. Ψ_Y is defined as follows:

$$\Psi_Y^p \triangleq \underset{t}{max} \mid\mid Y - T \mid\mid_p^{OD} \tag{3.8}$$

where T = {t$_1$,...,t$_d$,...,t$_D$} is an arbitrary vector with the same specifications of $\hat{Y}$ mentioned in relation:

$$\hat{y}_{m,j} = \begin{cases} 1, & \text{j = d} \\ 0, & \text{otherwise} \end{cases} \tag{3.9}$$

In E^p_{NOD} ordinal distance is used to take the order of categories into account while also normalizing along the largest possible ordinal. This normalization ensures the ordinal distance becomes trivial to interpret. They conclude by showing this newly introduced ordinal distance's performance and advantages using a number of numerical examples.

These related articles give valuable insight into the similarity measures we are performing, and also give context to our work in relation to the existing scientific literature. We acknowledge that this is more specific than what our novel introduction have use for, but nevertheless can be an interesting extension of our scientific contribution for future work.

3.3 Transformation Techniques

In this scientific literature search we are looking for useful introductions or definitions of techniques that help us transform our ordinal categorical data into numerical data. This was mentioned as trivial in several articles, but the lack of any definition or explanation of how to implement it was surprisingly not present in the literature we encountered. In contrast to the existing papers we investigated, our use is very narrow and specific in that we want to *learn* an *existing* transformation relationship between ordinal and numerical data when the domains of both these spaces are already defined and populated. Nevertheless we present here a few articles that research similar concepts of what we aim to do.

Harwell and Gatti [2001] show in their article how *Item Response Theory (IRT)* can be used to rescale ordinal data to an interval scale. This is advantageous because it allows arithmetic operations like addition/subtraction and multiplication/division to be applied allowing for trivial distance computations. They, like us, also commented on the observation that rescaling (transformation) in the context of *IRT* is lacking in educational research. It is emphasized that the complexity of applying IRT transformation models involves rigorous assumptions where failure to satisfy these assumptions makes using IRT inadvisable. To rescale ordinal data using IRT it is necessary to perform the following four steps:

1. Identify an appropriate IRT Model

2. Estimate Item Parameters
3. Estimate Proficiency Parameters
4. Assess Model-Data Fit

Their approach is complex and narrow in use, and goes to show that transforming ordinal data is no trivial task. It does not cover our need because it approaches the problem from a statistical stand point, rather than a *machine learning* stand point that we require.

Zdravevski et al. [2015] propose transformation based on the *Weight of Evidence* (WoE) parameter for nominal-to-numerical transformation. They also note that transformations of nominal and categorical data are not extensively researched, and they additionally highlight that the conventional way of utilizing nominal features is by converting them into a binary membership vector. This is expensive and complex because of the linear relationship between the size of the resulting vector and the size of the domain of the categorical features. Additionally, in the context of calculating distance in k-NN it is equivalent to performing the simple *overlap* similarity. Their experiments show a reduction in memory complexity, increase in accuracy and a shortening of execution time. Their approach is extensive and the reader is directed to their paper to implement this transformation. In any case, they generalize the original WoE to overcome some limitations that applies to it. Their solution is not applicable to our problem because the numerical domain that the nominal values are transformed to has to initially be empty. In addition they don't take advantage of the order that are present in our *ordinal* categorical features.

Chapter 4

Methodology

In this chapter we will present the architecture and design used to implement our demo and to run our scientific experiments and analyses. In Section 4.1 the experimental plan will be unveiled, and towards the end of the section we will talk about what behaviour and output we expect from our tests. We will thoroughly explain which steps we have taken in each stage of our development, as well as the decisions behind them. We will also include what experiments we have planned, and which Research Questions the experiments aim to answer. In Section 4.2 we will introduce the data we have used, all parameters, and how we generated it from our initial data. Thereby, the experiment setup and procedures will be explained in detail as to allow for these same experiments to be replicated and repeated for scientific peer reviews.

4.1 Experimental Plan

Recall from the introduction in Chapter 1 that we are going to handle mixed input and perform prediction in a working demo. We will implement transformation methods to be able to compare attributes of different data types, and we will compare various distance metrics, both categorical and continuous, with respect to performance accuracy. In terms of similarity, we are looking at similarity in relation to the retrieval step of CBR as mentioned earlier. We have decided to measure similarity with the supervised learning algorithm k-NN because it is simple to implement, straightforward to use, reliable in terms of performance, and directly encompasses the concept of similarity (or rather distance, which is considered the inverse of similarity). To take it a step further we have also implemented a simple, yet effective algorithm to train feature weights to gain insight into which attributes have more influence on the prediction, as to enable us to better make an educated decision for attribute selection when engaging in data transformation and similarity measures.

Our first step was to implement k-NN and tune its parameters to achieve a satisfactory level of accuracy. Initially we used a uniform weight distribution with a combined weight of 1. With this simple setup as default, we looped through plentiful of parameter combinations to observe different performance for individual parameter configurations, as well as the accumulated performance with respect to each parameter configuration. We tried out odd k-values ranging from 1 to 19, 5 different distance functions, and 6 different voting methods. Note that we only chose odd k-values as to guarantee no ties when voting. This of course only applies because our target class is binary. A sampling of the top performing parameter configurations can be observed in Table 4.1[1] and Table 4.2. With this information we went for some tuned parameters that are empirically more accurate (although the performance variations were minimal) before proceeding to training weights to learn which attributes have a higher significance with the chosen parameters. Keep in mind that we perform this on the original quantitative data set so that the achieved performance are considered to be the potential accuracy for when we perform similar tests on our diluted data[2] in our final experiments. Contrary to normal procedure, we employ *leave-one-out cross-validation* during our various iterations of preparation and experiments. This is because k-NN is an exception to the general workflow for building supervised machine learning models. This is to say k-NN is not *trained*, rather all the data is kept and used at run-time. The model created by k-NN is just the available labeled data placed in a metric space, unlike traditional machine learning models where there is a training step when building the model. After this we generated the new data set to be used in our experiments by using a *fuzzy logic* approach when translating the numerical values into categorical values. Furthermore we implemented and incorporated categorical similarity measures into our existing distance measure, as well as transformation techniques to handle mixed data types, and finally different similarity measure policies. Relevant parts of the final code can be viewed and inspected in Appendix B.

With this approach we hope to get valuable insight into the overall success of implementing a working demo for combined data types in accord with *RQ2*, our second research question. This will be evaluated based on: (1) the overall accuracy we achieve and the accuracy compared to our initial accuracy on the original quantitative data, and (2) a conversation about observations, limitations and considerations. Our first research question is already introduced and inves-

[1]Some performance metrics are omitted to fit the table, address Appendix A for further reading.

[2]By 'diluted' we mean that there is a loss in information contained by the data after preparing and modifying it to simulate our mixed data.

k	voting	distance	correctly	erroneously	accuracy
11	uniform	Euclidean	681	18	0.9742
9	uniform	Euclidean	680	19	0.9728
13	uniform	Euclidean	680	19	0.9728
15	uniform	Euclidean	680	19	0.9728
3	majority	Manhattan	679	20	0.9714

Table 4.1: Top 5 individual (unweighted) parameter configurations.

voting method	accuracy
logarithmic	0.9644
inverse	0.9645
uniform	0.9649
distance	**accuracy**
minkowski_3	0.9627
minkowski_2	0.9658
minkowski_1	0.9660
k	**accuracy**
9	0.9646
15	0.9648
19	0.9654

Table 4.2: Top 3 accumulated parameter configurations.

tigated in Chapter 2.3, and *RQ3* will be evaluated in our experiments based on performance accuracy when comparing categorical vs numerical similarity measures. We will perform a series of experiments aimed to answer our research questions, especially *RQ2* and *RQ3*. Our experiments will consist of running through multiple cycles of training (of both transformation methods and weight training) and with different combinations of policies of similarity measures and transformation methods. All three research question will be discussed accordingly in the following chapter.

From our experimental results we expect the following. With respect to *RQ2* we expect a significant drop in accuracy when carrying out our experiments on the mixed data compared to measuring similarity using the k-NN algorithm on the original quantitative data. We suspect the number of erroneous predictions will more than double. But the overall accuracy will not be unacceptable. We expect an accuracy for the prediction on the mixed data to lie somewhere between 80% and 90%. Regarding the performance of categorical vs numerical similarity measures and the choice of similarity policy, we expect a significantly greater performance when transforming to quantitative values, and using continuous similarity measures. With this, we mean to say that between the different policies we employ, the policy that favors the use of quantitative data and continuous similarity measures in most cases will have the highest accuracy.

4.2 Experimental Setup

The original data set used is from the open UCI Machine Learning Repository as introduced in Section 1.3. This data set consists of 699 instances of breast cancer occurrences classified as either 'benign' or 'malignant'. Each record has 11 attributes, one of which is the id number, 9 the feature variables, and the last one is the class attribute (2 corresponds to benign, and 4 malignant). All in numerical values, and the feature variables normalized between 0 and 10. In Table 4.3 the attributes are listed and described in an orderly fashion. For a data scientist the *description* column is not of immediate importance and can be discounted, unless a particular interest or a profound curiosity is present in the reader. The *domain* column is more relevant for the experiments we will run. In any case the table as a whole gives insight into the data we are using.

For our purpose the data is required to be a mix of numerical and categorical values. To achieve this we therefore generated a new data set from the original one, simulating this through a clever algorithm using fuzzy set logic. First we generated an entirely new data set consisting of only categorical values for the feature variables. The algorithm we designed to do this iterated over each feature variable of each record assigning it a generic categorical value (*level0*, *level1*, *level2*, *level3*) according to a fuzzy rule depicted in Figure 4.1. Basically what

attribute	**description**	**domain**
sample code number	id number	$\mathbb{N}$
class	binary target class stating benignity or malignity	2, 4
clump thickness	assesses if cells are mono- or multi-layered	0 - 10
uniformity of cell size	evaluates the consistency in size of the cells in the sample	0 - 10
uniformity of cell shape	estimates the equality of cell shapes and identifies marginal variances	0 - 10
marginal adhesion	quantifies how much cells on the outside of the epithelial tend to stick together	0 - 10
single epithelial cell size	relates to cell uniformity, determines if epithelial cells are significantly enlarged	0 - 10
bare nuclei	calculates the proportion of the number of cells not surrounded by cytoplasm to those that are	0 - 10
bland chromatin	rates the uniform texture of the nucleus in a range from fine to coarse	0 - 10
normal nucleoli	determines whether the nucleoli are small and barely visible or larger, more visible, and more plentiful	0 - 10
mitoses	describes the level of mitotic (cell reproduction) activity	0 - 10

Table 4.3: Attribute information of original data set.

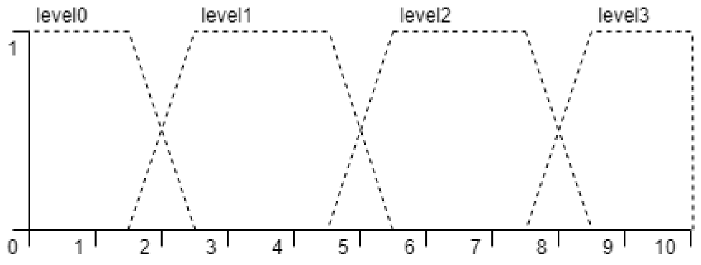


Figure 4.1: Fuzzy logic approach for translating data from numerical values to categorical values.

the algorithm does is to translate the numerical value into the corresponding categorical value of the box of which its numerical value lies within on the x-axis. In the case of overlapping boxes a stochastic function determines which categorical value the variable will take. That is, for each potential categorical value a number between 0 and the y-value of the respective box is randomly assigned, and the target categorical value is determined by the highest rolled random number between the boxes. For instance if the value 2.2 were to be translated to a categorical value (either 'level0' or 'level1'), a number would be randomly generated between 0 and 0.3 for *level0*, and likewise for *level1* a number would be generated randomly between 0 and 0.7. Repeating this example would obviously result in a majority of 'level1' translations, but there would still occur some 'level0' translations. Due to the inherited randomness, this is deemed sufficient to encapsulate the messiness that our real world entails in our generation of categorical values from numerical values. After generating this data set of categorical values we combined the numerical and categorical data sets to get a mixed one. This was done simply by choosing a ratio value, 0.5 in our case, and randomly copying either the corresponding categorical value into the original data set if a randomly generated number between 0 and 1 exceeded our ratio value. The code snippet for this data translation can be viewed in Appendix B.

Our experiments were built around testing different parameters configurations, transformation techniques and similarity measure policies to perform classification and determine the strength and effectiveness of said variations by our performance measures. The steps we took in the process of conducting our experiments were as follows:

1. Train weights for each parameter configuration.
2. Select parameter configurations.

k-value	5, 11
voting method	new_standard
continuous distance metric	manhattan, euclidean
categorical similarity measure	overlap

Table 4.4: The different parameter configurations used in our tests.

3. Run our k-NN classification algorithm using the leave-one-out method with:
    - Different transformation methods
    - Different similarity measure policies
4. Collect results and repeat from Step (2) with new parameter configurations.

In the first step we trained the feature weights for each of the parameter configurations mentioned in Section 4.1 on the original quantitative data. This was done by iteratively adjusting each attribute weight and performing classification on the data set until a performance peak was found, similarly to how our *Incremental Step* transformation method works. As weight training is not the focus of this project this approach was deemed adequate even though this method could only guarantee a local maximum. This almost halved the number of errors across the different parameter configurations. One would think that the attribute weights across different parameter configurations would tend to gravitate towards similar weight distributions—to an actual objective importance of each variable. But it was interesting to see how attributes with high and low importance for one parameter configuration could swap for the next. We note that this can be because of overfitting (due to a relative small number of data records), that our simple k-NN algorithm doesn't capture the proper causation that determines the class, or any other unforeseen cause.

In step two we chose the best performing parameter regarding the voting method; namely 'new_standard', the competing top performing continuous distance metrics; Manhattan and Euclidean distance, and finally the top performing k-value of 11. In k-NN, choosing a too low k-value may result in overfitting, and likewise a too high k-value likely results in underfitting. The rule of thumb is to be in the ballpark of $\log(samplesize)$, and the k-value we found to be best coincides with this. Overfitting and underfitting in k-NN also depends on the *level of noise* in samples and number of classes. In retrospect we also ran the experiments on the k-value of 5 to get some variation over this parameter as well. When we ran our tests we made sure to use the particular weights trained for the parameter configuration in question as to have an equal playing field when comparing results up against each other. The parameter configurations we used in our tests can be seen in Table 4.4. Note that we needed a way to convert the categorical similarity measure to a distance metric (to be compatible with the distance computation in k-NN). Normally this is done by the self-explanatory Formula 4.1.

$$similarity = \frac{1}{1 + distance} \tag{4.1}$$

Although this has the inherit inconvenience that calculating the distance from a similarity of 0 is beyond the bounds of possibility because of division by zero. However, since our algorithm measures the distance of each attribute independently, our conversion from similarity to distance is reasonably done by exploiting the property that our attributes are normalized between 0 and 10. Formula 4.2 and Formula 4.3 below encapture this:

$$similarity = 1 - \frac{distance}{10} \tag{4.2}$$

$$distance = 10(1 - similarity) \tag{4.3}$$

In this implementation the similarity, that is 1 for matches and 0 for mismatches, is correctly translated into minimum distances of 0 for similarity values of 1 and maximum distances of 10 for similarity values of 0. Following, our similarity measure assign a similarity value between two data instances X and Y belonging to the data set as follows:

$$S(X, Y) = \sum_{k=1}^{d} w_k S_k(X_k, Y_k) \tag{4.4}$$

Here S is the similarity function, X and Y are the data instances, d the data variables, w_{k} is the attribute weight for the k^{th} variable, and X_{k} and Y_{k} are the k^{th} variable for the data instances X and Y, respectively.

For step three we simply ran the k-NN classification algorithm according to the different parameter configurations in Table 4.4 on the techniques in Table 4.5 that we are comparing with respect to the research questions. The explanations for the different similarity measure policies can be seen in Table 4.6. For our proposed transformation method *Incremental Step* we trained the transformation method on each individual parameter configuration to get a tailored mapping table with a customized and optimized performance. The cycle of our experimental procedure is depicted in the flowchart in Figure 4.2.

similarity measure policy	all_cont, mixed_cont, mixed_cat, all_cat
transformation method	incremental_step

Table 4.5: The different techniques we are comparing against each other.

policy / **data types**	**all_cont**	**mixed_cont**	**mixed_cat**	**all_cat**
both continuous	quantitative measure	quantitative measure	quantitative measure	qualitative measure
mixed types	quantitative measure	quantitative measure	qualitative measure	qualitative measure
both categorical	quantitative measure	qualitative measure	qualitative measure	qualitative measure

Table 4.6: Explanation for the different similarity measure policies we test.

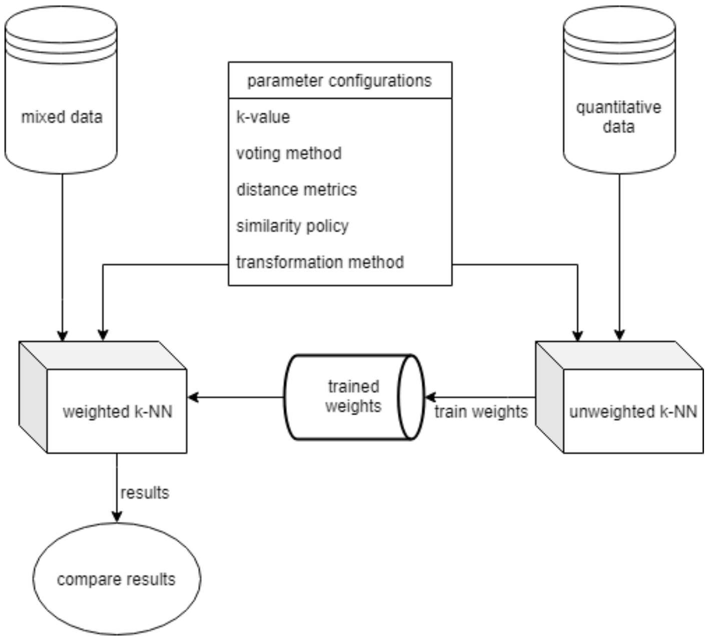


Figure 4.2: An overview of the procedural flow in our experiments.

Chapter 5

Results and Discussion

In this chapter we will present the results in an orderly way, visualized with graphical tools, and make statistical observations in Section 5.1. We will evaluate said results and share some thoughts in Section 5.2, and finally discuss merits and limitations of our work and give remarks to our results with the research questions in mind in Section 5.3.

5.1 Results

In our work we have used a data set of clinical breast cancer diagnostic cases. In this data set there were two classifications we were predicting, namely malignant and benign cases of breast cancer. There were 241 and 458 instances of each class, i.e. malignant and benign respectively—totaling 699 instances as shown in Table 5.1. As can be seen in the same table, this makes roughly $^1/_3$ of the instances malignant and $^2/_3$ benign. This is a circumstantially fair distribution in terms of class representation.

Furthermore, we ran our classification algorithm to predict these classes comparing categorical against continuous similarity measures using different policies

	instances	**percentage**
benign	458	65.5%
malignant	241	34.5%
total:	699	100%

Table 5.1: Breast Cancer data set class distributions.

k	cont	policy	cor	err	acc	pre	rec	f-1
5	min_1	all_cont	675	24	.966	.972	.976	.974
5	min_1	all_cat	659	40	.943	.947	.967	.957
5	min_1	mix_cont	655	44	.937	.935	.972	.953
5	min_1	mix_cat	640	59	.916	.905	.974	.938
5	min_2	all_cont	674	25	.964	.972	.974	.973
5	min_2	all_cat	649	50	.928	.936	.956	.946
5	min_2	mix_cont	658	41	.941	.948	.963	.956
5	min_2	mix_cat	644	55	.921	.909	.978	.942
11	min_1	all_cont	680	19	.973	.972	.987	.979
11	min_1	all_cat	652	47	.933	.938	.961	.949
11	min_1	mix_cont	648	51	.927	.936	.954	.945
11	min_1	mix_cat	617	82	.883	.918	.902	.910
11	min_2	all_cont	673	26	.963	.974	.969	.972
11	min_2	all_cat	661	38	.946	.957	.961	.959
11	min_2	mix_cont	668	31	.956	.967	.965	.966
11	min_2	mix_cat	650	49	.930	.927	.969	.948

Table 5.2: Results of prediction, comparing different similarity measure policies on different parameter configurations.

to elect in which cases we used which similarity measures. As covered in Chapter 4, this was done with different parameter configurations in order to compare performance under different circumstances. Our main results are listed in Table 5.2. In this table the header labels are the following labels shortened (in order): *k-value*, *continuous similarity measure*, *similarity measure policy*, *(N° of) correctly classified instances*, *(N° of) erroneously classified instances*, *accuracy*, *precision*, *recall*, and *f_1-score*. Also, min_1 and min_2 stands for $Minkowski_1$ (Manhattan distance) and $Minkowski_2$ (Euclidean distance). For the meaning of the terms in the **pol** (similarity measure policy) column, refer to Table 4.6 in Chapter 4.2. For all of these tests we used the *voting method* 'new_standard', and the *Overlap* categorical similarity measure. The *Incremental Step* transformation method is used across the board. These techniques were introduced in Chapter 2. From Table 5.2 it can be seen that the similarity policy 'all_cont' performs better on all accounts except where, surprisingly, 'mixed_cat' has a higher performance score with regards to the *recall* measure when the parameter distribution is *k*-value = 5 and the Euclidean distance is used.

In Table 5.2 the results are very dense and it is hard to immediately and easily extract useful information. Therefore a bar graph that simplifies the results and

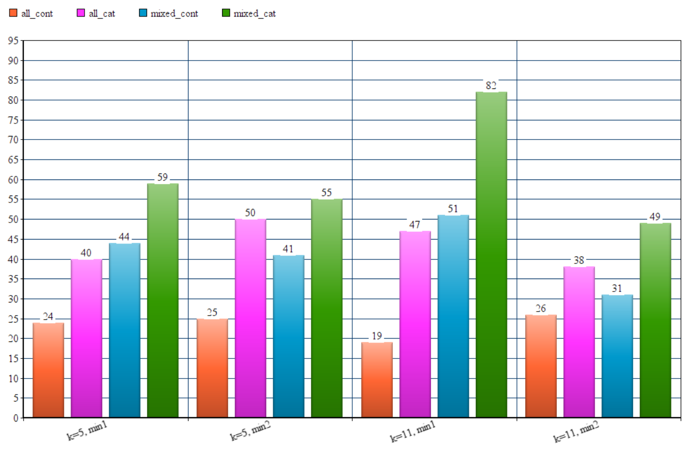


Figure 5.1: Bar graph visualizing error distributions.

visualizes the performance can be seen in Figure 5.1. Since the number of erroneously classified instances is small relative to the total number of instances, we are showing the number of erroneously classified instances to better represent the relative performance difference. The results are grouped into four groups, each group showing the performance for a unique parameter combination. The red, pink, blue, and green column indicate the number of errors for the *All Continuous*, *All Categorical*, *Mixed Continuous*, and *Mixed Categorical* similarity measure policy, respectively (from left to right). From this graph it can be observed with respect to accuracy that: (1) the *All Continuous* similarity measure policy is performing better than all the others, and (2) the *Mixed Categorical* similarity measure policy performs worst within each individual group. It can also be seen that even the worst performance of the *All Continuous* similarity measure policy performs better than the best performing non-*All Continuous* similarity measure policy among all groups.

In Figure 5.2 we have separately visualized the four different performance measures: *accuracy*, *precision*, *recall*, and *f_1-score* of the similarity measure policies across the parameter configurations. Figure 5.2a shows the accuracy, Figure 5.2b the precision, Figure 5.2c the recall, and Figure 5.2d the f_1-score. All these graphs seem to reflect each others behaviour, except for the *recall* graph where the *Mixed Categorical* similarity measure policy suddenly exhibits some unusual characteristics compared to the other three graphs. Namely it is sharing the top performance with the *All Continuous* similarity measure policy, until suddenly it drops momentarily for the parameter configurations k-value $= 11$ and

performance / policy	correct	error	accuracy	precision	recall	f1-score
all_cont	675.5	23.5	0.9665	0.9725	0.9765	0.9745
all_cat	655.25	43.75	0.9375	0.9445	0.9613	0.9528
mixed_cont	657.25	41.75	0.9403	0.9465	0.9635	0.955
mixed_cat	637.75	61.25	0.9125	0.9148	0.9558	0.9345

Table 5.3: Mean performance for the different similarity measures policy.

Minkowski$_1$.

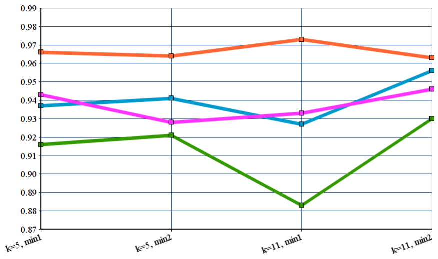
Following we have computed the mean performance for each of the similarity measure policies by the accumulated parameter configurations. This is shown in Table 5.3. It can be seen in this table that the mean performance of each similarity measure policy is strictly better or strictly worse than their counterparts across all performance measure. Namely:

$$all\_cont > mixed\_cont > all\_cat > mixed\_cat \tag{5.1}$$

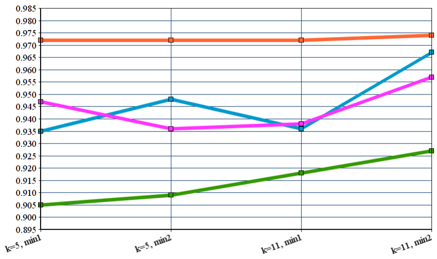
Furthermore, this is neatly visualized in Figure 5.3. Here each performance metric is grouped together and compared between each similarity measure policy.

5.2 Evaluation

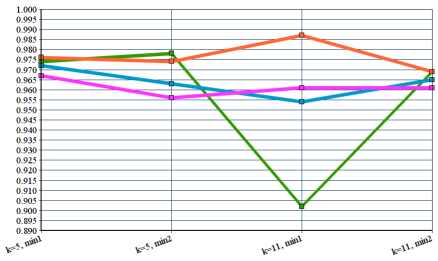
We had a data set with 699 classified instances of breast cancer occurrences. As Table 5.1 shows there is an acceptably even distribution of classes with 241 instances being classified *malignant* and 458 instances classified *benign*. With this fairly even class distribution it is not possible to have one class poorly classified without this being reflected in the plain *accuracy* performance metric. In general for medical prediction the *recall* performance measure is the most important, because it is the measure that tells us how many of the patients with the sickness we actually are able to correctly identify. Thus allowing for further testing if many healthy people also got classified with the sickness. *Recall* alone can be misguiding on its own in the case all instances are classified as malignant, because then the *recall* would actually be 100% even though the prediction algorithm gives absolutely no useful new information. That's why *recall* should be observed in tandem with the *f_1-score*. However, as our choice of data set is arbitrary and we only wish to academically measure the objective usefulness regardless of which data set is used, we will mostly consider the *accuracy* score together with the *f_1-score* to ensure no class predictions are unevenly bad.



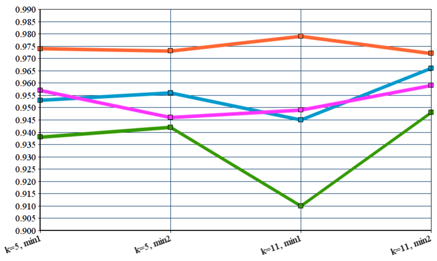
(a) Accuracy



(b) Precision



(c) Recall



(d) F_1-score

Figure 5.2: Accuracy, Precision, Recall, F_1-score of results over the different parameter distributions.

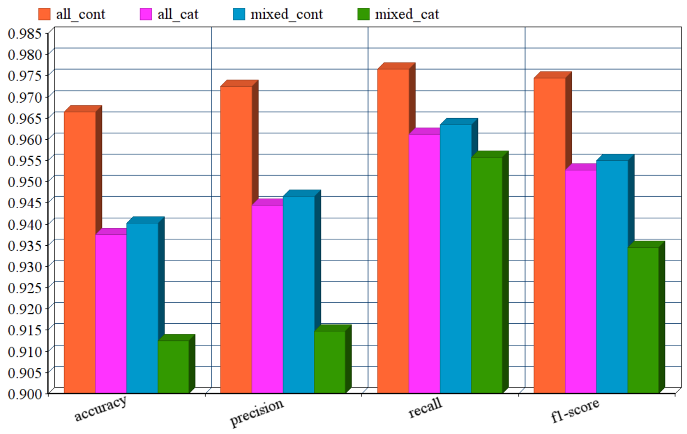


Figure 5.3: Bar graph visualizing mean performance.

In the graph in Figure 5.1 the error distribution is shown across the four parameter configurations. As mentioned in Section 5.1 the *All Continuous* similarity measure policy compares better, and the *Mixed Categorical* similarity measure policy compares worse, for each parameter configuration grouping. If we calculate the mean and variance over the parameter configurations as shown in Table 5.4, we see that the *variance* for the parameter configuration k-value $= 11$ and the Manhattan distance is less stable than the other parameter configurations. At the same time, the opposite behaviour is exhibited by the Euclidean distance for the same k-value. This indicates that with a higher k-value the performance fluctuates more depending on the continuous distance metric used. This is an interesting observation as Minkowski$_1$ (Manhattan) distance places equal importance on attribute distances, while Minkowski$_2$ (Euclidean) punishes when individual attributes have a greater distance. This indicates that the high variance mentioned earlier can be caused by imprecise distances resulting from an inaccurate transformation method. Thus, inaccurate distances that get abnormally large are not properly punished by the Manhattan distance. This is more evident for higher values of k because more nearest neighbors are included that increases the chance of including these abnormalities. The same behaviour is seen for k-values of 5, but on a much smaller scale. This is of course merely speculations—can be *correlation* and not *causation*.

In Table 5.5 it can be seen that the *All Continuous* similarity measure policy

k	distance	mean	variance
5	minkowski$_1$	41.75	155.2
5	minkowski$_2$	42.75	130.2
11	minkowski$_1$	49.75	498.7
11	minkowski$_2$	36	74.5

Table 5.4: Mean and variance over parameter configurations.

sim. meas. policy	mean	variance
all continuous	23.5	7.25
all categorical	43.75	24.19
mixed continuous	41.75	51.69
mixed categorical	61.25	156.19

Table 5.5: Mean and variance over similarity measure policy.

outperforms the other similarity measure policies. Following, in terms of mean accuracy performance we have *All Mixed Continuous*, *All Categorical*, and lastly *Mixed Categorical*. This is illustrated in the first group of the graph in Figure 5.3. The same order is also evident for the other performance measures as can be seen in the same graph. It seems that the similarity measure policy that favors transformation to numerical values to then perform continuous similarity measure tends to have a better performance, but the two worst policies contradict this tendency—namely *All Categorical* and *Mixed Categorical*. Although *Mixed Categorical* uses continuous similarity measures for when both instances are numerical, it still is outperformed heavily by *All Categorical* that translates both numerical values into categorical ones before performing categorical similarity measures on them. So if the apparent rule was that the preference is to always use categorical similarity measures or always use continuous similarity measures on homogeneous values, then *All Categorical* should have scored higher than *Mixed Continuous*. That is not the case either. The performance of both *All Categorical* and *Mixed Continuous* are very close to each other, and which performs better actually depends on the parameter configuration.

From the tendencies of the performance mean in Table 5.5 there are two dominant behaviours that can be drawn: (1) is that continuous similarity measures are preferred to categorical similarity measures, and (2) not translating homogeneous values is also preferred. This explains the similar performance of *All Categorical* and *Mixed Continuous* because they both satisfy one of the behaviours that are preferred. This can be seen in Table 5.6. In this matrix better performance is toward the upper left corner, and worse performance is toward the lower right corner.

This behaviour is also clearly visible in three of the four performance measures graphed in Figure 5.2. Figure 5.2a, Figure 5.2b, and Figure 5.2d all clearly show *All Continuous* as the top performer, *All Categorical* and *Mixed Continuous* with a similar performance, and *Mixed Categorical* with the worst performance. Figure 5.2c exhibits odd behaviour for the *Mixed Categorical* measure. This shows that for three of the four parameter configurations *Mixed Categorical* performs

		continuous preferred	
		✓	✗
no translation	✓	all cont	all cat
	✗	mixed cont	mixed cat

Table 5.6: Matrix of preferred behaviour.

on par with *All Continuous* with regards to classifying benign cases as benign. It is hard to imagine that it boils down to coincidental randomness, but at the same time the behaviour depicted in Table 5.6 is evident for three out of the four parameter configurations. What makes this even more bizarre is that for the fourth parameter configuration, it actually drops down to performing significantly worse than all three other policies. This odd behaviour directly contradicts the preferred behaviour observation in Table 5.6 we drew from the results in Table 5.5. However as seen in Figure 5.3, *recall* is still the worst performer on average for *Mixed Categorical* as well because the *recall* performance drop for one of the parameter configurations is so substantial.

5.3 Discussion

The evaluation of our results demonstrate that the *All Continuous* similarity measure policy clearly yields the best performance. Our initial expectations, mentioned in Section 4.1, that the similarity measure policies that prefer continuous similarity measures would turn out superior was only partially correct though. While the absolute strongest method was *All Continuous*, the *All Categorical* performed better than *Mixed Categorical* even though the *Mixed Categorical* employs continuous similarity measures for homogeneous numerical values. We also said we expected the number of errors to more than double. In Table 5.7 we list the number of errors on the quantitative vs mixed data set with regards to the same parameter configurations. We see that on two accounts the number of errors more than doubled, although surprisingly just barely. For the other

k	cont. sim. measure	quantitative data set	mixed data set
5	Minkowski$_1$	14	24
5	Minkowski$_2$	12	25
11	Minkowski$_1$	11	19
11	Minkowski$_2$	12	26

Table 5.7: Errors with quantitative data against mixed data.

two parameter configurations the number of errors did not double, although on all four accounts the number of errors was very close to doubled. The pattern here is that with regards to relative performance between quantitative data and mixed data, the Minkowski$_1$ distance has a percentagewise smaller increase in number of errors compared to Minkowski$_2$ distance. Our final prediction was that the *accuracy* would fall to between 80% and 90% on the mixed data set. Although this was a fairly uninformed estimate, we are positively surprised to see that we achieved an average accuracy of a whopping 96.7% for *All Continuous*. The average of all the similarity measure policies were all above 90% as seen in Table 5.3. Only in one individual case, which can be seen in Table 5.2, did the *accuracy* drop below 90%, and that was for the *Mixed Categorical* similarity measure policy for the parameter configurations: k-value = 11 and Minkowski$_1$ distance. From Table 5.3 it can be calculated that the number of errors of *All Continuous* is approximately 44%, 46% and 62% lower than the number of errors for respectively *Mixed Continuous*, *All Categorical* and *Mixed Categorical*.

There are a few limitations to our work. In our attempt to compare similarity measures we used a simple, but time consuming classification algorithm that limited the number of test runs we could afford to carry out. On a similar note our solution has very many parts that could be tweaked and tuned to empirically test more variations—each part with considerably complex optimisation challenges. Lastly, we used an *nominal* approach for the categorical similarity measure where, for our data set, a *ordinal* approach could be used. With this we didn't fully take advantage of the ordinal property of our data, however this was compensated with our transformation method that learned this *ordinal* relationship in the data. In our experiments it is also a considerable challenge to identify in which of the many parts in our implementation there exists a potential for improvement. For instance if we employed an unsuitable categorical similarity measure, it could negatively impact the validity of our results. Taking all this into account, we are nevertheless satisfied that our efforts have introduced a novel problem area into the scientific community where there now is a lot of potential for advancements.

5.3.1 Research Questions

Lastly we will conclude this chapter by summarizing and commenting our findings in the context of our Research Questions.

Research Question 1 *What transformation techniques exist to handle input parameters of both quantitative and qualitative form?*

Because of the novelty of the problem we are working with, useful techniques in existing research were not found. We accept this failure to implement multiple transformation techniques, but leave this topic for future work in this field. We did however define a simple working technique that got the job done—that can be used as a basis or comparison for future alternative techniques.

Research Question 2 *How can suitable techniques be implemented in a working demo that accepts and handles input parameters of both quantitative and qualitative form?*

We have employed various techniques including *continuous* and *categorical similarity measures*, and different *similarity measure policies* to implement a working demo – with surprisingly acceptable results – that handles mixed data types. We implemented a weighted k-NN classification algorithm that utilizes either Manhattan and Euclidean distance with the *Incremental Step* transformation method to achieve a staggering 96.7% accuracy for mixed, heterogeneous data.

Research Question 3 *How do techniques for using categorical vs numerical measures, based on the training data, compare with respect to performance differences?*

As reported in Section 5.1 and Section 5.2 we found that the *All Continuous* similarity measure policy clearly outperforms the alternatives with regards to either of the performance measures *accuracy*, *precision*, *recall*, and *f_1-score*. In Table 5.6 we speculate which behaviour is preferred with regards to *similarity measure policy*. This was shown with both the Euclidean distance and the Manhattan distance compared to the Overlap similarity. The average number of errors were reduced by 44% - 62% for the *All Continuous* similarity measure policy depending on which alternate policy it is compared to.

Chapter 6

Conclusion

In this chapter our study is concluded by summarizing our contributions and proposing future work. Section 6.1 lists our contributions in the research area of *similarity measures* on heterogeneous attributes, while in Section 6.2 we propose the natural next steps to further advance the research field we have been working in.

6.1 Contributions

Our overarching contribution has been the introduction of a novel area of research into the scientific community. Although our study has only been of a shallow nature with regards to the different sub-areas it has touched upon, we are confident our contributions constitutes a meaningful initial step on uncharted territory. Our main contributions are listed below:

- We have introduced a novel research problem in this respective field, particularly in the context of similarity measures in CBR. However, also applicable outside the scope of CBR.
- We have built a working demo as a proof of concept with respect to similarity measures on data with heterogeneous attribute types by transforming between categorical and numerical values.
- Finally we have evaluated and compared performance of different strategies that uses categorical contra continuous similarity measures when confronted with mixed data types on classification tasks.

6.2 Future Work

With regards to future work in the domain of this novel problem area, there is frankly a lot of potential advancements waiting to be made. We were surprised to find so little existing literature on this specific topic, and because of this we were unable to meet our initial expectations that we had for this project. Consequently we invite other researchers and scientists to continue exploring this topic. Specifically we propose the three directions listed below as natural next steps:

1. We hope our contributions will motivate further investigation, implementation and comparisons of transformation techniques to allow for mutual translation between categorical and continuous data. Both for *nominal* and *ordinal* data types. A minor improvement with regards to the transformation is to consider the merging of categories that have negligible impact over the classifications. However, this should not be dependent on the parameter configurations.

2. We welcome iterations of improvements to our proposed *Incremental Step* transformation method.

3. Because of the novelty of our work, we encourage the generation of more empirical data by running further experiments with new categorical/continuous similarity measures, weight learning methods, and especially on other machine learners than k-NN, etc.

Bibliography

Aamodt, A. and Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Commun.*, 7(1):39–59.

Bahari, M. and Van hamme, H. (2014). *Normalized Ordinal Distance; A Performance Metric for Ordinal, Probabilistic-ordinal or Partial-ordinal Classification Problems*, pages 285–302.

Boriah, S., Chandola, V., and Kumar, V. (2008). Similarity measures for categorical data: A comparative evaluation. volume 30, pages 243–254.

Cheung, Y.-M. and Jia, H. (2013). Categorical-and-numerical-attribute data clustering based on a unified similarity metric without knowing cluster number. *Pattern Recogn.*, 46(8):2228–2238.

Cunningham, P. (2008). A taxonomy of similarity mechanisms for case-based reasoning. *IEEE Transactions on Knowledge and Data Engineering*, 21(11):1532–1543.

Han, J., Kamber, M., and Pei, J. (2012). *Data Mining: Concepts and Techniques.* Morgan Kaufmann.

Harwell, M. R. and Gatti, G. G. (2001). Rescaling ordinal data to interval data in educational research. *Review of Educational Research*, 71(1):105–131.

Lewis, N. (2019). Would you trust an algorithm to diagnose an illness? https://edition.cnn.com/2019/07/15/business/artificial-intelligence-healthcare. Accessed: July 15, 2019.

Mitchell, T. M. (1997). *Machine Learning.* McGraw-Hill Science/Engineering/-Math.

Norvig, P. and Russell, S. J. (1994). *Artificial Intelligence: A Modern Approach.* Prentice Hall.

Richter, M. M. and Weber, R. O. (2013). *Case-Based Reasoning: A Textbook*. Springer-Verlag Berlin Heidelberg.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3).

Schank, R. C. (1983). *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*. Cambridge University Press.

Stahl, A. (2005). Learning similarity measures: A formal view based on a generalized cbr model. In Muñoz-Ávila, H. and Ricci, F., editors, *Case-Based Reasoning Research and Development*, pages 507–521, Berlin, Heidelberg. Springer Berlin Heidelberg.

Wikipedia (2019). Fuzzy logic. https://en.wikipedia.org/wiki/Fuzzy_logic. Accessed: July 30, 2019.

Willems, S. J., Albers, C. J., and Smeets, I. (2019). Variability in the interpretation of dutch probability phrases - a risk for miscommunication. arXiv:1901.09686v1.

Wolberg, D. W. H. (1992). Breast cancer wisconsin (original) data set. https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(original). Accessed: February 11, 2019.

Yala, A., Lehman, C., Schuster, T., Portnoi, T., and Barzilay, R. (2019). A deep learning mammography-based model for improved breast cancer risk prediction. *Radiology*, 292(1):60–66. PMID: 31063083.

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, 8(3):338–353.

Zdravevski, E., Lameski, P., Kulakov, A., and Kalajdziski, S. (2015). Transformation of nominal features into numeric in supervised multi-class problems based on the weight of evidence parameter. In *2015 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 169–179.

Appendices

A Detailed results and data

Top and bottom 10 unweighted parameter configurations

All the 300 parameter configurations that were tested are too many to include, but below the top 10 and bottom 10 (in terms of accuracy) is shown.

k	**voting**	**distance**	**correct**	**false**	**accuracy**	**precision**	**recall**	**f1-score**
11	uniform	minkow_2	681	18	0.9742	0.9846	0.9760	0.9803
9	uniform	minkow_2	680	19	0.9728	0.9824	0.9760	0.9792
13	uniform	minkow_2	680	19	0.9728	0.9824	0.9760	0.9792
15	uniform	minkow_2	680	19	0.9728	0.9824	0.9760	0.9792
3	majority	minkow_1	679	20	0.9714	0.9824	0.9738	0.9781
3	inverse	minkow_1	679	20	0.9714	0.9824	0.9738	0.9781
3	logarithmic	minkow_1	679	20	0.9714	0.9824	0.9738	0.9781
5	uniform	minkow_1	679	20	0.9714	0.9803	0.9760	0.9781
7	inverse	minkow_2	679	20	0.9714	0.9803	0.9760	0.9781
7	uniform	minkow_1	679	20	0.9714	0.9803	0.9760	0.9781
3	standard	minkow_9	663	36	0.9485	0.9587	0.9629	0.9608
3	standard	minkow_{10}	663	36	0.9485	0.9587	0.9629	0.9608
3	new_standard	minkow_9	663	36	0.9485	0.9587	0.9629	0.9608
3	new_standard	minkow_{10}	663	36	0.9485	0.9587	0.9629	0.9608
5	new_standard	minkow_9	663	36	0.9485	0.9587	0.9629	0.9608
5	new_standard	minkow_{10}	663	36	0.9485	0.9587	0.9629	0.9608
9	new_standard	minkow_9	663	36	0.9485	0.9587	0.9629	0.9608
9	new_standard	minkow_{10}	663	36	0.9485	0.9587	0.9629	0.9608
11	new_standard	minkow_9	663	36	0.9485	0.9587	0.9629	0.9608
11	new_standard	minkow_{10}	663	36	0.9485	0.9587	0.9629	0.9608

'minkow' is short for the minkowowski distance.

B Python code

Code snippet for Fuzzifier for categorical data generation

```
def fuzzifier(value):
    v = int(value)
    classes = ['thin', 'normal', 'thick', 'very_thick']
    class1 = random.uniform(0, min(1, max(0, 1.5 - v)))
    class2 = random.uniform(0, min(min(max(-1.5 + v, 0), 1),
                                   max(min(5.5 - v, 1), 0)))
    class3 = random.uniform(0, min(min(max(-4.5 + v, 0), 1),
                                   max(min(8.5 - v, 1), 0)))
    class4 = random.uniform(0, min(max(-7.5 + v, 0), 1))
    winner = max(class1, class2, class3, class4)
    if winner == class1:
        return classes[0]
    elif winner == class2:
        return classes[1]
    elif winner == class3:
        return classes[2]
    else:
        return classes[3]
```

Code snippet for Incremental Step transformation learner

```
def learn_mapping_structure():
    global mapping_table, k, parameters

    for record in breast_cancer_data:
        for index, attribute in enumerate(record[2:]):
            if type(attribute) == str and attribute not in \
                    mapping_table[0].get(
                        file_reader.headers[index + 2], []):
                mapping_table[0][
                    file_reader.headers[index + 2]][
                    attribute] = 5.0

    min_step = 0.1
    categories_ordered = sorted(list(enumerate(weights)),
                                key=lambda x: x[1])

    parameters['policy'] = 'all_cont'
    for variable in categories_ordered:
        variable = file_reader.headers[variable[0] + 2]
        for domain in mapping_table[0][variable]:
            print(variable + ':␣' + domain)
            possibles = [float(i) for i in range(11)]
            for x in possibles:
                mapping_table[0][variable][domain] = x
                false = 0
                for test_object in breast_cancer_data:
                    prediction = k_nearest_neighbor(
                        breast_cancer_data, test_object)
                    if prediction != test_object[1]:
                        false += 1
                possibles[int(x)] = false
            print(possibles)
            new_candidates, falses = [i for i in
                                      range(len(possibles))
                                      if
                                      possibles[i] == min(
                                          possibles)], [
                                         possibles[i]
                                         for i in
                                         range(len(
                                             possibles))
                                         if
                                         possibles[
                                             i] == min(
                                             possibles)]

            final_candidates = []
            for index, candidate in enumerate(
                    new_candidates):
                mapping_table[0][variable][
                    domain], false = candidate, falses[
                    index]
                lowest, turned, limit_hit, step, accumulated_step_adjustme
                while len(limits) < 2:
                    old_value = mapping_table[0][variable][
    domain]
mapping_table[0][variable][
    domain] = min(max(0,
                      mapping_table[0][
                          variable][
                          domain] + step),
                  10)

new_false = 0
for test_object in breast_cancer_data:
    prediction = k_nearest_neighbor(
        breast_cancer_data, test_object)
    if prediction != test_object[1]:
        new_false += 1

print(
    mapping_table[0][variable][domain])
print(new_false)

if new_false > false:
    mapping_table[0][variable][
        domain] = old_value
    if not turned:
        step, accumulated_step_adjustments = −1
        turned = True
    else:
        limit_hit = True
        if abs(step / 2) < min_step:
            if false < lowest:
                limits.pop()
            limits.append(
                mapping_table[0][
                    variable][domain])
            lowest, step, accumulated_step_adjust
            limit_hit = False
        else:
            step, accumulated_step_adjustments =
else:
    if not turned and new_false < false:
        turned = True
    if abs(step / 2) < min_step or \
                    mapping_table[0][variable][
                        domain] == 0 or \
                    mapping_table[0][variable][
                        domain] == 10:
                turned = True
                if false < lowest:
                    limits.pop()
                limits.append(
                    mapping_table[0][variable][
                        domain])
                lowest, step, accumulated_step_adjustments =
                limit_hit = False
            elif limit_hit:
                step, accumulated_step_adjustments = step / 2
            false = new_false
    final_candidates.append(
        (false, limits.copy()))

final_candidates, counter, min_candidate = sorted(
    final_candidates, key=lambda x: x[0]), 0, 0
print(final_candidates)
for cand in final_candidates:
    if cand[0] == final_candidates[0][0]:
        min_candidate += (cand[1][0] + cand[1][
            1]) / 2
        counter += 1
min_candidate /= counter

dist, min_cand2 = 10, None
for cand2 in final_candidates:
    if cand2[0] == final_candidates[0][0]:
        if abs((cand2[1][0] + cand2[1][
            1]) / 2 - min_candidate) <= dist:
            dist, min_cand2 = abs((cand2[1][0] +
                                   cand2[1][
                                       1]) / 2 - min_candidat
                              cand2[1]

mapping_table[0][variable][domain] = (min_cand2[
                                          0] +
                                      min_cand2[
                                                                            1]) / 2
            print(str(
                mapping_table[0][variable][domain]) + '\n')

    for key in file_reader.headers[2:]:
        for i in range(11):
            smallest_distance = None
            for cat, value in mapping_table[0][key].items():
                if abs(
                        i - value) < smallest_distance or smallest_di
                    mapping_table[1][key][i] = cat
                    smallest_distance = abs(i - value)

    with open('mapping.csv', 'w') as new_mapping_file:
        new_mapping_file.write(
            str(mapping_table[0]).replace('[{', '').replace(
                '}]', '') + '\n')
        new_mapping_file.write(
            str(mapping_table[1]).replace('[{', '').replace(
                '}]', ''))
```